Save T-Rex

Chrome Dino-bot AI

Group 2

Gavin Rowsell
Sravan Pingali
Srikanth Chowdary Thumati
Umutcan Asutlu

Introduction

With the advent of high performance processors and excellent communication systems, we are rapidly seeing the implementation of AI in almost every industry. These technological improvements have made it possible for AI to be powered in daily used consumer products.

But there has been one industry which has been using them for many decades now. That is the video game industry. In the 1980s, these AI were rudimentary but games developed now house some of the most complex AI systems implemented in them, however they are still powered by the basic systems introduced in the 80's.

So our team decided that games would be the perfect field for us to get started with AI development. Our projects goal is to develop a self-learning AI that will play the beloved dino game of the chrome browser and eventually score get a high score in it.

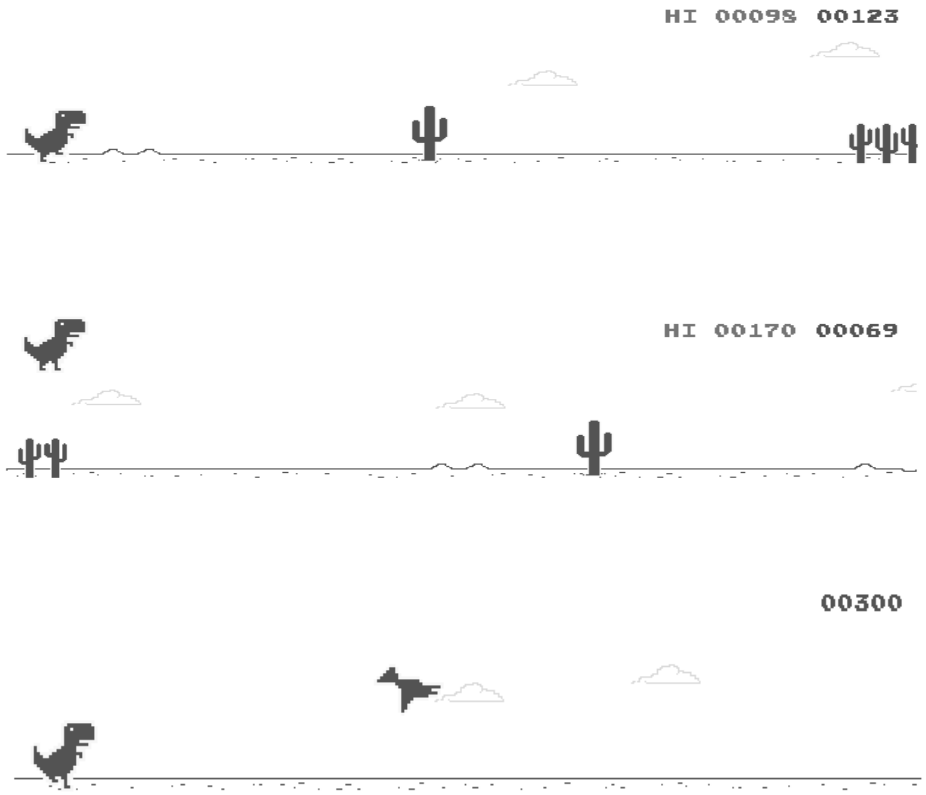
How we perceive the game

In this simple game environment humans see a dinosaur moving endlessly to the right as obstacles continuously appear to block the path.

The further the dino moves to the right, the more points the player gets and the faster the dino moves.

In order to continue the game they must use the spacebar to jump over the obstacles in front of them.

As the game progresses, obstacles will start appearing in the top half of the screen (sky) rather than the bottom (ground), if the player jumps as they did with previous obstacles, they will make contact with them ending the game.

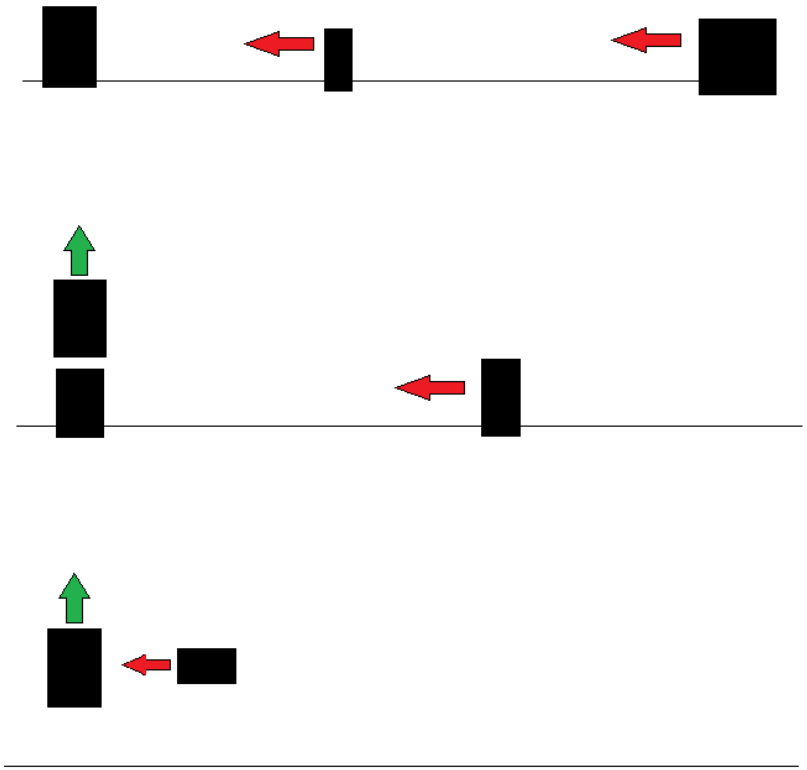


How a computer perceives the game

A computer on the other hand would interpret the game in its most basic form: entities moving to the left towards a second entity which can be moved up and down.

With reinforcement learning, the model will gradually learn to move the player entity up (jump) as the obstacle entity approaches it.

With this approach, during the training process we can expect the model to always initially make the mistake of jumping into the airborne obstacle since it will be behaving off its past experience to jump when an entity approaches the player.



Reinforcement learning approach

For the model to perform better there will be 3 major milestones it will need to achieve in the training process.

1. Learn that the model will be punished if it does not jump when an entity reaches a certain x - distance from the player
2. Learn that the model will be punished if it jumps into an entity on the top half of the screen (airborne obstacles)
3. Learn that the specific x distance to perform a jump will increase over time (game speeds up therefore jumps have to be executed earlier)

   The optimal model will learn the specific rate of change of the x distance from an obstacle to perform a jump.

Deep Q Learning and MLP

From our research we were able to find that Q-learning and MLP models have good performance.

Deep Q learning learns to play the game by successfully reading the pixels without the need for any feature extraction. Hence, it usually performs better than methods that use feature extractions.

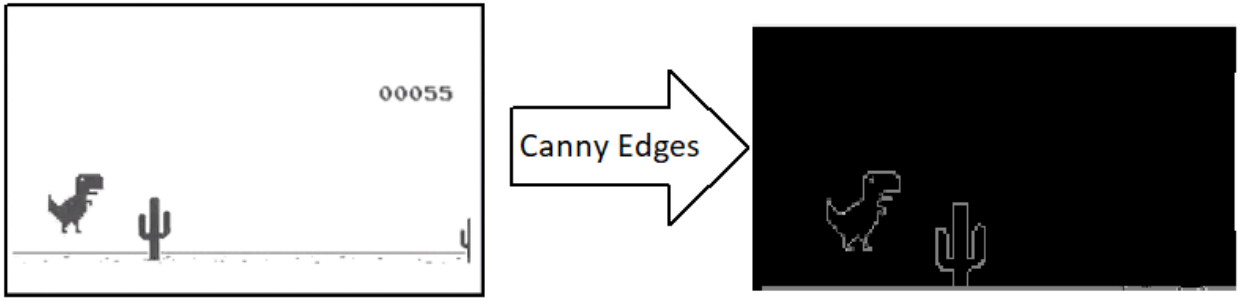
MLP on the other hand helps in refining the parameters by experience.

However both of these methods have difficulty with the handling velocity.

Computer Vision

Computer Vision will be used by the team to detect objects as they approach the dinosaur.

CNN will be used to object detection and feature extraction of objects such as edges.



Orignal Image

Edges Detected

Storing Progress

Our trex bot will learn from its mistakes and improve itself to get better score every time it plays the game.

Hence, we will develop our system in a way that it will also track the bot’s learning process. This includes the score bot achieves every game and the obstacle it failed.

The data will be reachable externally.

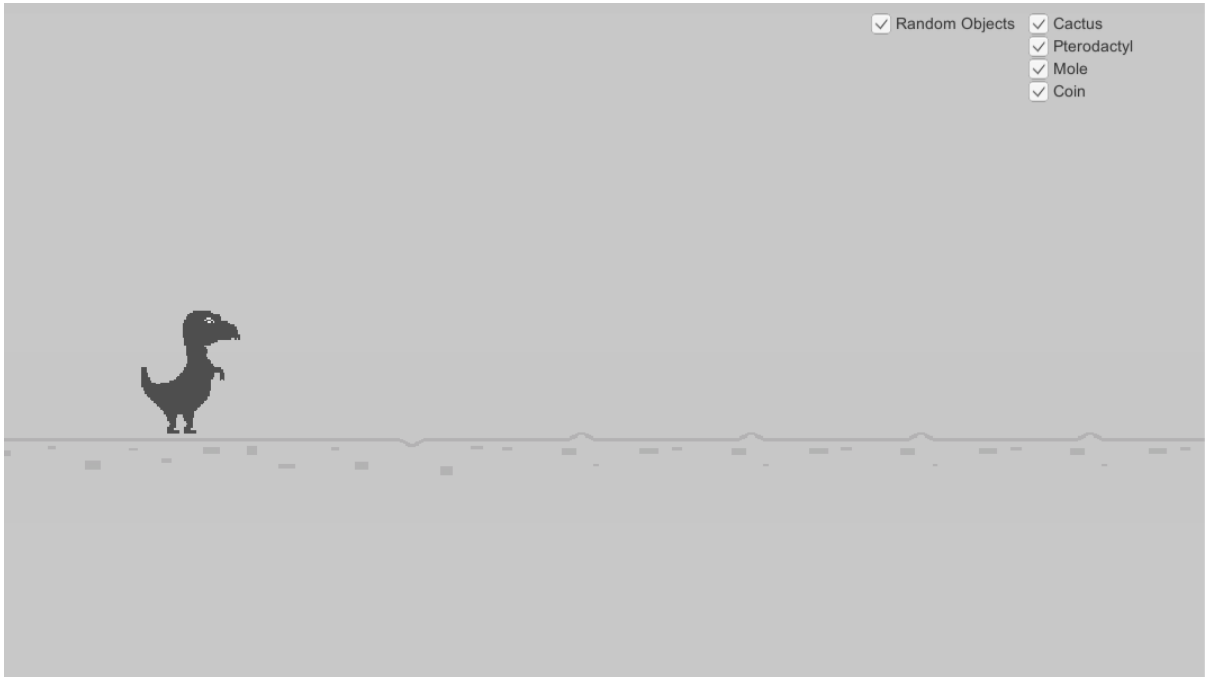
Interactable Training

Our clone of the chrome dino game will allow the user to select which objects will randomly appear on screen.

The cactus and pterodactyl already exist in the original game however two new game objects will be added.

The coin will award points, the AI will have to learn that not every approaching entity needs to be avoided.

The mole will be an obstacle that the AI has to “scare” away by jumping before it reaches the dino.



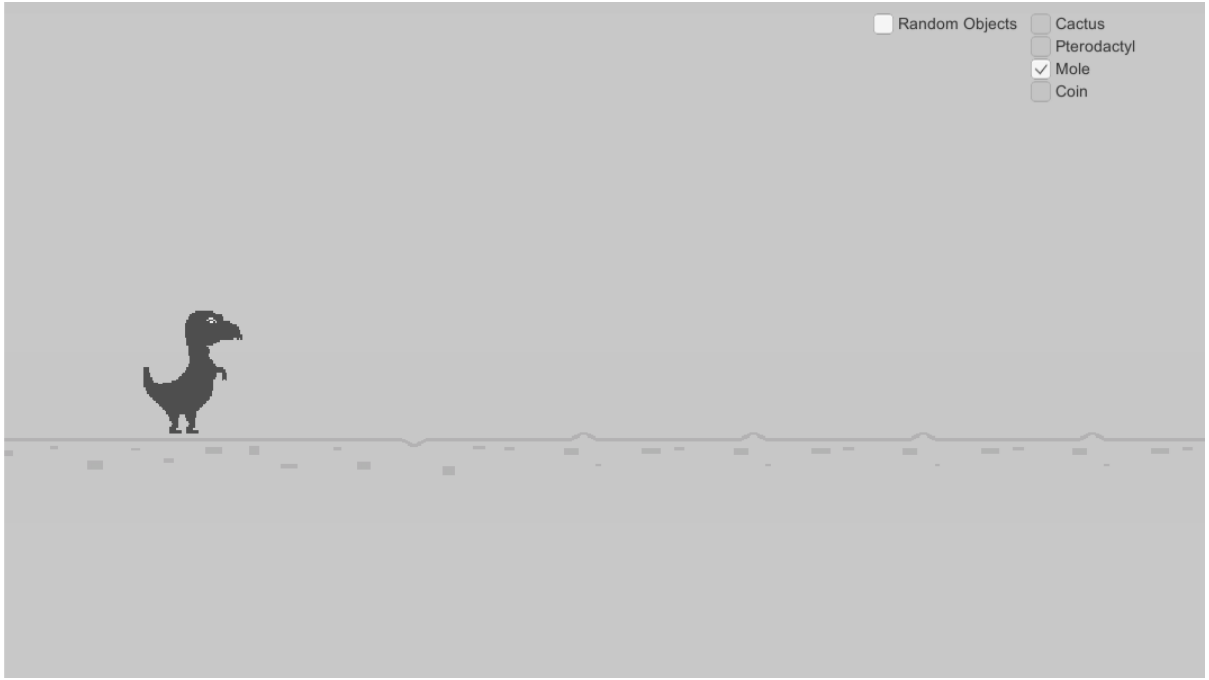
Interactable Training

By selecting and deselecting certain objects the user will be able to see how the AI learns to respond to different elements in the game.

Additionally, if the user deselects the random field they will be able to manually spawn in objects through key presses.

This will give the user full control over the learning environment to see how the AI will react in specific scenarios.

Ex. Spawn only coins at first then spawn cactus to see how many attempts before it learns to avoid the new object.



Technologies Used

Unity will be used to develop the chrome dino clone with proposed additional game mechanics.

Game will be a standalone application initially, nice to have would be to deploy the app on a website along with python app (AI model) for ease of access.

Tensorflow will be used to create, train and maintain the AI model (CNN).

Python code will run in tandem with the Unity app to capture, process, then feed images to the model.

Project Goals

The end goal would be to present users with the chance to experience first hand how a computer vision AI learns to react in a controlled environment over time.

A potential expansion to the project would be to include the colour channels (RGB) to the game and AI model so a comparison of performance can be made.

This way users could see how the two models differ from each other.

I.E. learning rate difference (number of attempts to learn new behaviour) or object recognition accuracy (coloured objects may be easier to distinguish from each other)

We can expect better performance from a model receiving more input (RGB vs grayscale) however training such a model may take more time.